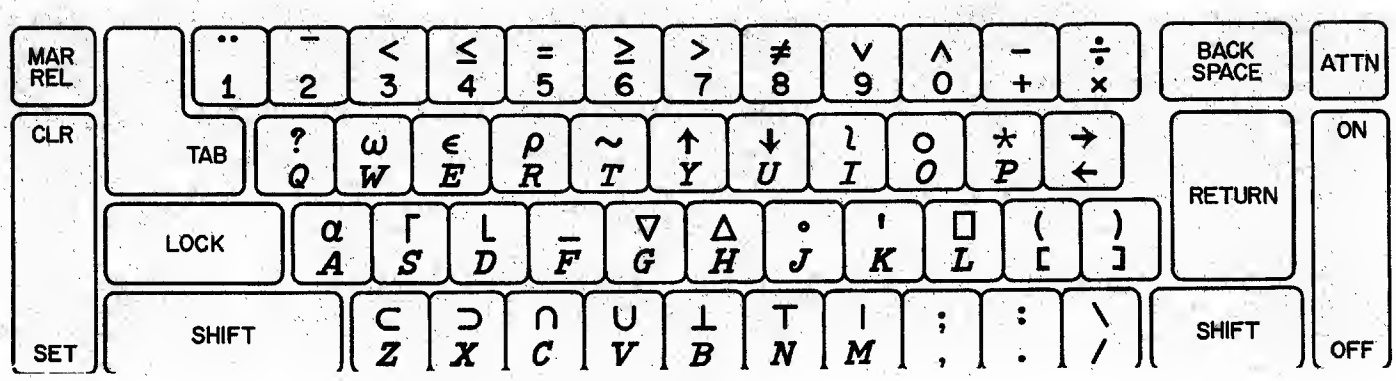
APL / 360

- LANGUAGE
- TIME-SHARING SYSTEM

A BRIEF DISCUSSION OF
SOME DISTINCTIVE FEATURES



THE APL\360 KEYBOARD

The APL\360 Language and Time-Sharing System were developed at the T. J. Watson Research Center, Yorktown Heights New York. In some respects, the language differs considerably from conventional programming languages. The objective of this pamphlet is to present a brief discussion of some of these differences. For a fuller discussion, the reader is referred to the works cited in the bibliography.

While the major part of this report is concerned with the ways in which APL is different from a language like FORTRAN, for example, it is also important to note points of similarity. A FORTRAN programmer can, in general, write an APL program in just the same way he would write a FORTRAN program, i.e. without using the special characteristics of APL which differ from FORTRAN. For example, the FORTRAN statements

```
DIMENSION A(5,5)
RETURN
```

are directly analogous to the APL\360 statements

```
A←5 5⍴0
→0
```

The differences between the FORTRAN and corresponding APL statements lie primarily in the greater generality of APL. For example, the APL equivalent of a 'DIMENSION' statement specifies not only the dimensions of the variable 'A', but also the values of its components. In the example given, this value is zero for all of them. Thus, the APL statement is a combination of the FORTRAN 'DIMENSION' and 'DATA' statements. The APL equivalent of a 'RETURN' statement is used not only to return from a subroutine but also to terminate a 'main-line' program. There is no distinction between the 'desk calculator' and 'program' mode. All programs are functions. Any function may be 'called' from the keyboard or by another function.

The APL\360 keyboard, shown in the frontispiece, displays the basic symbols used in the language. In general, each of the shift characters corresponds to one or more APL primitive functions. In addition, some functions are denoted by compound symbols, e.g.⍟ ,! for the logarithm and factorial functions.

To highlight a few of the important differences between APL and other programming languages, the following pages will be devoted to the topics of:

input and output
reduction
compression and expansion
inner, or scalar, or generalized matrix product
outer product

INPUT AND OUTPUT IN APL

INPUT DIRECTLY FROM A TERMINAL IS MERELY THE BASIC APL\360 SPECIFICATION STATEMENT. FOR EXAMPLE, THE STATEMENT

```
A←3 3⍴4 2 1.7 45 9 12 1 2 3
```

STORES A 3×3 ARRAY, CALLED 'A', WHOSE COMPONENTS, IN ROW LIST ORDER, ARE THE NINE NUMBERS FOLLOWING THE SYMBOL '⍴'.

A REQUEST FOR OUTPUT IS EVEN SIMPLER. THE EXECUTION OF ANY APL EXPRESSION WHICH IS NOT PRECEDED BY A NAME AND A SPECIFICATION SYMBOL,'←', IS IMMEDIATELY FOLLOWED BY A DISPLAY OF THE CALCULATED VALUE. THUS, THE BASIC 'OUTPUT' STATEMENT CONSISTS MERELY OF TYPING THE NAME OF THE VARIABLE WHOSE OUTPUT IS DESIRED OR OF SPECIFYING THE CALCULATION TO BE PERFORMED. TO DETERMINE THE VALUE OF A VARIABLE 'A', ALL THAT IS REQUIRED IS THE FOLLOWING:

```
      A

 4    2    1.7
45    9   12
 1    2    3
```

THE MATRIX VALUES SHOWN ABOVE WERE PRODUCED BY THE COMPUTER AS OUTPUT. FORMATTING,(INTEGER,LOGICAL,REAL,CHARACTER), IS AUTOMATIC.

IF INPUT IS NOT DIRECTLY FROM THE KEYBOARD, BUT IS TO BE REQUESTED BY A PROGRAM, AN INPUT-OUTPUT SYMBOL,'⍞', IS USED AS THE RIGHT-HAND PART OF A SPECIFICATION STATEMENT. IN A PROGRAM, THE STATEMENT:

```
A←⍞
```

WOULD BE WRITTEN TO REQUEST INPUT AT THAT POINT IN THE PROGRAM. WHEN PROGRAM EXECUTION REACHED THIS STATEMENT, THE PROGRAM WOULD WAIT FOR INPUT FROM THE TERMINAL KEYBOARD. WHEN INPUT WAS COMPLETE, (I.E. WHEN THE CARRIAGE RETURN KEY WAS DEPRESSED), THE INPUT WOULD BE STORED AND IDENTIFIED BY THE NAME 'A'. THE INPUT COULD BE A SCALAR, A VECTOR, A MANY-DIMENSIONED ARRAY, A CHARACTER STRING OR ARRAY, OR ONE OF SEVERAL OTHER POSSIBILITIES.

SINCE MORE THAN ONE SPECIFICATION STATEMENT IS PERMITTED IN A SINGLE LINE, THE INPUT STATEMENT CAN BOTH STORE A VALUE AND PROVIDE AN ARGUMENT FOR A FUNCTION. THE STATEMENT

```
A←+/⍞
```

STORES ONLY THE SUM OF THE INPUT VALUES (THE ROW SUM IF THE INPUT IS A MATRIX).

THE STATEMENT

```
A←+/B←⍞
```

STORES THE INPUT VALUES AS 'B' AND THEIR SUM AS 'A'.

COMPONENT-BY-COMPONENT OPERATIONS, REDUCTION AND THE ELIMINATION OF 'DO' LOOPS

IN CONVENTIONAL PROGRAMMING LANGUAGES, MANY REPETITIVE OPERATIONS ARE MOST CONVENIENTLY PERFORMED WITH 'DO' LOOPS OR THEIR EQUIVALENTS. IN MANY CASES, THIS SEQUENTIAL KIND OF PROGRAMMING IS NOT NEEDED IN APL.

COMPONENT-BY-COMPONENT OPERATIONS. IF WE WISH TO MULTIPLY THE COMPONENTS OF A VECTOR CALLED 'QUANTITIES' BY THE CORRESPONDING COMPONENTS OF A VECTOR CALLED 'PRICES',THE APL STATEMENT TO DO IT WOULD BE

```
EXTENSIONS←QUANTITIES×PRICES
```

AS A RESULT OF THIS CALCULATION, THE VECTOR 'EXTENSIONS' WOULD BE THE VECTOR OF COMPONENT-BY-COMPONENT PRODUCTS. NOTHING ANALOGOUS TO A 'DO' LOOP WOULD BE NEEDED.

IF WE HAD ONE TAX RATE WHICH APPLIED TO ALL THESE PRODUCTS AND WE WISHED TO CALCULATE A VECTOR OF INDIVIDUAL TAXES BY ITEM, THE STATEMENT

```
ITEMTAXES←TAXRATE×EXTENSIONS←QUANTITIES×PRICES
```

WOULD DO THIS FOR US, IN ADDITION TO DOING THE MULTIPLICATIONS DESCRIBED ABOVE.

THE RULE IS THAT ANY DYADIC FUNCTION DEFINED FOR SCALARS IS CARRIED OUT COMPONENT-BY-COMPONENT FOR COMPATIBLY DIMENSIONED OPERANDS. IF ONE OF THE OPERANDS IS A SCALAR, IT WILL BE TREATED AS AN ARRAY OF IDENTICAL COMPONENTS WHICH HAS THE SAME DIMENSIONS AS THE OTHER OPERAND. IN OTHER WORDS, SCALAR MULTIPLICATION OF ANY ARRAY IS POSSIBLE, AND SO ARE SCALAR ADDITION, SUBTRACTION, DIVISION, EXPONENTIATION, AND SO ON.

REDUCTION. ANOTHER IMPORTANT PLACE IN WHICH APL ELIMINATES THE NEED FOR 'DO' LOOPS IS IN THE 'REDUCTION' OF A VECTOR TO A SCALAR, A MATRIX TO A VECTOR, AND SO ON. A COMMON EXAMPLE OF SUCH REDUCTION IS THE ADDITION OF ALL THE COMPONENTS OF A VECTOR TO GET A SCALAR SUM, OR MULTIPLICATION OF THE COMPONENTS TO GET A SCALAR PRODUCT. THESE OPERATIONS ARE CUSTOMARILY SYMBOLIZED BY THE GREEK LETTERS Σ AND Π IN MATHEMATICS OR STATISTICS TEXTS. THEY WOULD ORDINARILY BE CALCULATED BY A ' DO ' LOOP IN FORTRAN.

IN APL, REDUCTION IS SYMBOLIZED BY A FUNCTION SYMBOL FOLLOWED BY A '/'.

```
+/ IS DIRECTLY EQUIVALENT TO Σ
×/ IS DIRECTLY EQUIVALENT TO Π
```

AS IS CHARACTERISTIC OF APL, ANY APPROPRIATE FUNCTION SYMBOL MAY PRECEDE THE '/'. FOR EXAMPLE, THE LOGICAL FUNCTIONS 'OR',(∨), AND 'AND',(∧), CAN BE USED TO REDUCE LOGICAL VECTORS. SO CAN SUCH FUNCTIONS AS 'MAXIMUM',(⌈), 'MINIMUM',(⌊), AND OTHERS.

COMPRESSION

'COMPRESSION' IS THE OPERATION OF SELECTING ELEMENTS FROM AN ARRAY. AN EXAMPLE, SHOWING THE COMPRESSION OF A STRING OF LETTERS MAY HELP MAKE THIS CLEAR.

```
      1 0 1 1 0 1/'ABCDEF'
ACDF
```

NOTE THE SELECTED LETTERS ARE THOSE WHICH CORRESPOND IN POSITION TO '1'S IN THE LOGICAL VECTOR PRECEDING THE SLASH.

COMPRESSION USES THE SAME SYMBOL ,'/', AS REDUCTION BUT PRECEDES IT WITH A LOGICAL VECTOR RATHER THAN A FUNCTION SYMBOL.

LOGICAL VECTORS CAN BE GENERATED BY USE OF THE RELATION FUNCTIONS. IF 'A' IS A VECTOR, '3<A' IS A VECTOR, WITH THE SAME NUMBER OF COMPONENTS AS 'A', BUT WITH ZEROES WHERE 'A' HAS COMPONENTS GREATER THAN OR EQUAL TO 3 AND ONES WHERE 'A' HAS COMPONENTS LESS THAN 3. IF 'A' IS A VECTOR OF INTEGERS, THE VECTOR '2|A', (THE VECTOR OF RESIDUES MODULO 2 OF THE COMPONENTS OF A) IS ALSO A LOGICAL VECTOR SINCE ITS ONLY COMPONENT VALUES ARE ZERO OR ONE. IT CAN BE USED TO COMPRESS 'A' AS FOLLOWS:

```
      A←3 4 5 9 12 16 17

      (2|A)/A
3  5  9  17
```

AND,OF COURSE, THE RESULT OF THE COMPRESSION CAN BE REDUCED, THUS

```
      ×/(2|A)/A
2295
```

GIVES US THE PRODUCT OF ALL THE ODD COMPONENTS OF 'A'.

EXPANSION

'EXPANSION' IS THE OPERATION INVERSE TO COMPRESSION. TO ILLUSTRATE WITH A CHARACTER STRING:

```
      1 0 1 1 0 1\'ACDF'
A CD F
```

AND WITH A NUMERICAL VECTOR:

```
      1 0 1 1 0 1\1 2 3 4
1  0  2  3  0  4
```

INNER PRODUCT. GENERALIZED MATRIX PRODUCT.

FOR THOSE UNFAMILIAR WITH THE MATHEMATICAL NOTION 'INNER' OR 'SCALAR' PRODUCT, IT CAN BEST BE INTRODUCED BY A RETURN TO THE QUANTITIES-PRICES EXAMPLE GIVEN ABOVE. IN CALCULATING A BILL, WE MULTIPLY QUANTITIES BY PRICES AND ADD THE RESULTS TO GET THE TOTAL AMOUNT OF THE BILL. THIS CAN BE DONE IN APL AS A COMBINATION OF COMPONENT-BY-COMPONENT MULTIPLICATION AND SUM REDUCTION.

```
      QUANTITIES←2 3 4 5
      PRICES←1 1.5 2 2.375
      +/QUANTITIES ×PRICES
26.375
```

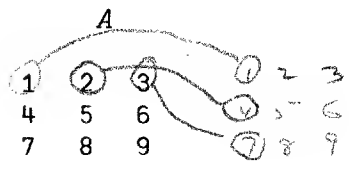
THE RESULT SHOWN AT THE LEFT IS THE COMPUTER-CALCULATED AMOUNT OF THE TOTAL BILL.

FOR MANY PURPOSES, IT IS USEFUL TO SYMBOLIZE THIS COMPONENT-BY-COMPONENT OPERATION COMBINED WITH A SUBSEQUENT REDUCTION OPERATION AS A SINGLE, COMPOUND OPERATION, SYMBOLIZED ,(IN THIS CASE), BY THE SYMBOLS '+.×',OR, IN GENERAL, BY THE REDUCTION OPERATION, A DOT, AND THE COMPONENT-BY-COMPONENT OPERATION. OUR BILL TOTAL, FOR EXAMPLE, COULD BE CALCULATED AS FOLLOWS:

```
      QUANTITIES+.×PRICES
26.375
```

THIS CONVENTION IS PARTICULARLY USEFUL WHEN WE WISH TO PERFORM MATRIX MULTIPLICATION, SINCE SUCH MULTIPLICATION IS MERELY THE CALCULATION OF THE INNER PRODUCT SHOWN ABOVE FOR ALL COMBINATIONS OF ROWS IN THE PREMULTIPLYING MATRIX WITH COLUMNS IN THE POSTMULTIPLYING MATRIX. AN EXAMPLE WILL ILLUSTRATE THIS:

```
      A←3 3⍴⍳9
      A
```



R1C1 R1C2 R1C3
30 36
66
102

```
      A+.×A
 30  36  42
 66  81  96
102 126 150
```

AGAIN, AS IS CHARACTERISTIC OF APL, ANY TWO DYADIC SCALAR FUNCTIONS CAN BE SUBSTITUTED FOR THE '+' AND '×' OF THE CONVENTIONAL MATRIX MULTIPLICATION.

```
      A⌈.+A
 10  11  12
 13  14  15
 16  17  18
```

THIS IS THE KIND OF CALCULATION USEFUL, FOR EXAMPLE, IN CRITICAL-PATH CALCULATIONS.

OUTER PRODUCT

AS WE HAVE SEEN, THE INNER PRODUCT OF TWO VECTORS RESULTS IN A SCALAR. CONVERSELY, THE OUTER PRODUCT OF TWO VECTORS FORMS A MATRIX.

THE MOST FAMILIAR EXAMPLE OF AN OUTER PRODUCT IS THE MULTIPLICATION TABLE. FOR EXAMPLE, TO CALCULATE A MULTIPLICATION TABLE OF THE FIRST FIVE INTEGERS, WE COULD AVAIL OURSELVES OF THE ⍳ FUNCTION WHICH, GIVEN AN ARGUMENT 'N', GENERATES A VECTOR OF INTEGERS FROM 1 TO N. USING '⍳5' AS BOTH PREMULTIPLIER AND POSTMULTIPLIER, THE CALCULATION OF A MULTIPLICATION TABLE WOULD BE DONE AS FOLLOWS:

```
      (⍳5)∘.×⍳5

 1  2  3  4  5
 2  4  6  8 10
 3  6  9 12 15
 4  8 12 16 20
 5 10 15 20 25
```

LIKE THE INNER PRODUCT, THE OUTER PRODUCT IS DENOTED BY A COMPOUND SYMBOL. IN THE OUTER PRODUCT, HOWEVER, THE REDUCTION FUNCTION SYMBOL IS REPLACED BY THE 'NULL' SYMBOL, '∘'. ANOTHER ILLUSTRATION MAY HELP FURTHER CLARIFY THE IDEA OF 'OUTER PRODUCT'. THE FOLLOWING FUNCTIONAL EXPRESSION CALCULATES THE SQUARES, CUBES, SQUARE ROOTS AND CUBE ROOTS OF THE FIRST FIVE INTEGERS:

```
      (⍳5)∘.*2 3,÷2 3

 1   1  1           1
 4   8  1.414213562 1.25992105
 9  27  1.732050808 1.44224957
16  64  2           1.587401052
25 125  2.236067977 1.709975947
```

THE OUTER PRODUCT IS EXTREMELY USEFUL IN THE 'PIVOTING' OPERATIONS CHARACTERISTIC OF MATRIX ALGEBRA.

A FINAL NOTE

A BRIEF DISCUSSION OF THIS TYPE CAN HARDLY DO MORE THAN SUGGEST A FEW OF THE FEATURES IN WHICH APL\360 DIFFERS FROM OTHER PROGRAMMING LANGUAGES. READERS INTERESTED IN EFFICIENT ALGORITHMIC LANGUAGES APPLICABLE IN A WIDE VARIETY OF ENDEAVORS ARE URGED TO CONSULT THE REFERENCES LISTED ON THE BACK COVER. READERS INTERESTED IN 'INTERACTIVE' TIME-SHARING SYSTEMS WILL ALSO FIND MANY OF THESE PUBLICATIONS TO BE OF INTEREST.

BIBLIOGRAPHY

Berry, P.C., APL\360 Primer, IBM Corporation, 1968.

Berry, P.C., APL\1130 Primer, IBM Corporation, 1968.

Breed, L.M., and R.H. Lathwell, "The Implementation of APL\360", ACM Symposium on Experimental Systems for Applied Mathematics, Academic Press, 1968.

Falkoff, A.D., and K.E. Iverson,"The APL\360 Terminal System", ACM Symposium on Experimental Systems for Applied Mathematics, Academic Press, 1968.

Falkoff, A.D., K.E. Iverson, and E.H. Sussenguth, "A Formal Description of System/360", IBM Systems Journal, Volume 3, Number 3, 1964.

Iverson, K.E., A Programming Language, Wiley, 1962.

Iverson, K.E., Elementary Functions: an algorithmic treatment, Science Research Associates, 1966.

Iverson, K.E., "The Role of Computers in Teaching", Queen's Papers in Pure and Applied Mathematics, Volume 13, Queen's University, Kingston, Canada, 1968.

Lathwell, R.H., APL\360: Operator's Manual, IBM Corporation, 1968.

Lathwell, R.H., APL\360: System Generation and Library Maintenance, IBM Corporation, 1968.

Pakin, S., APL\360 Reference Manual, Science Research Associates, 1967.

Rose, A.J., Videotaped APL Course, IBM Corporation, 1968.

Smillie, K.W., Statpack 1: An APL Statistical Package, Publication No. 9, Department of Computing Science, University of Alberta, Edmonton, Canada, 1968.

SAMPLE EXECUTION OF AN APL\360 PROGRAM

SEVERAL EXECUTIONS OF AN APL\360 PROGRAM CALLED 'GOETZ' ARE GIVEN BELOW TO ILLUSTRATE HOW A USER-WRITTEN PROGRAM IS EXECUTED FROM THE KEYBOARD. THE LINES IN WHICH THE WORD 'GOETZ' APPEARS ARE REQUESTS FOR EXECUTION OF THE PROGRAM FOR THE INDICATED VALUES. THE LINES DIRECTLY BELOW THESE ARE THE COMPUTED OUTPUT.

WHAT IS THE FUNCTION 'GOETZ?'

```
      GOETZ 3
ARGUMENT TOO LARGE

      GOETZ 2
ARGUMENT TOO LARGE

      GOETZ 1
1

      GOETZ .9
2  3  15

      GOETZ .8
2  4  20

      GOETZ .85
2  3  60

      GOETZ .456
3  9  87  16313  532211622

      GOETZ .787
2  4  28  778  2723000

      GOETZ .6
2  10

      GOETZ .5
2

      GOETZ .2
5
```